IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A method **for prevention of priority inversion without priority promotion mechanism when** ~~of~~ executing processes with different priorities in a multiprocessing environment comprising execution of a low priority process and a high priority process **and optionally at least one intermediate priority process having a priority between that of the low priority process and the high priority process**, where the high priority process ~~(T4)~~ and the low priority process share a given resource, ~~and wherein the multiprocessing environment includes real-time and non-real-time operating systems,~~ the method comprising the step of:

temporarily raising an effective priority of the low priority process when the low priority process is going to use the shared resource, ~~where the effective priority is raised to be above a priority of an other process in the multiprocessing environment~~ **wherein the step of raising the effective priority comprises executing/assigning an additional process accessing the shared resource on behalf of the low priority process where the additional process has a priority higher than the at least one intermediate priority process in the multiprocessing environment**.

2. (Currently Amended) Method according to claim 1, wherein the step of raising the effective priority comprises the steps of: executing/assigning an additional process ~~(T3, T5)~~ accessing the shared resource ~~(SM4, 402')~~ on behalf of the low priority process ~~(T1)~~ where the additional process ~~(T3, T5)~~ has a priority equal to the effective priority, and where the additional process is synchronised with the low priority process.

3. (Currently Amended) Method according to claim 1, wherein the multiprocessor environment comprises a real-time operating system and a non-real time operating system running on a single processor at least at a given time, where the real-time

operating system comprises said high priority **process** ~~thread~~ and said additional process and where the non-real time operating system comprises said low priority **process** ~~thread~~.

4. (Previously Presented) Method according to claim 2, wherein the additional process and the low priority process are synchronised using a first semaphore and a second semaphore.

5. (Currently Amended) Method according to claim 1, wherein the effective priority is raised at least until

the low priority process has accessed or used the shared resource, or

the high priority process has accessed or used the shared resource if the high priority process ~~(T4)~~ attempts to access or use the shared resource while the low priority process has access or uses the shared resource.

6. Cancelled.

7. (Previously Presented) Method according to claim 4, wherein access to the shared resource is controlled by a mutex (M) whereby said additional process will not wait for the low priority process as long as it owns the mutex (M).

8. (Previously Presented) Method according to claim 1, wherein the shared resource is selected from the group of:

a shared memory,

a shared file, and

a shared input/output (I/O) device.

9. (Currently Amended) Method according to claim 1, **wherein** ~~characterized in that~~ the high priority process executes time-critical tasks.

10. (Currently Amended) A system for **prevention of priority inversion without priority promotion mechanism, the system** executing processes with different priorities

in a multiprocessing environment comprising means adapted to execute a low priority process and a high priority process **and optionally at least one intermediate priority process having a priority between that of the low priority process and the high priority process,** where the high priority process and the low priority process share a given resource~~, and wherein the multiprocessing environment includes real-time and non-real-time operating systems~~, the system comprises:

a processor for temporarily raising an effective priority of the low priority process ~~(T1)~~ when the low priority process ~~(T1)~~ is going to use the shared resource, ~~where the effective priority is raised to be above a priority of an other process in the multiprocessing environment.~~ **wherein the processor executes an additional process accessing the shared resource on behalf of the low priority process where the additional process has a priority higher than the at least one intermediate priority process in the multiprocessing environment.**

11. (Previously Presented) System according to claim 10, wherein a processor for raising the effective priority is conceived to:

executing an additional process accessing the shared resource on behalf of the low priority process where the additional process has a priority equal to the effective priority, and

where the system comprises synchronisation means conceived to synchronise the additional process with the low priority process.

12. (Currently Amended) System according to claim 10, wherein the system comprises the processor comprises a real-time operating system and a non-real time operating system running on the processor at least at a given time, where the real-time operating system comprises said high priority **process** ~~thread~~ and said additional process and where the non-real time operating system comprises said low priority **process** ~~thread~~.

13. (Previously Presented) A computer readable medium having stored thereon instructions for causing one or more processing units to execute **a** ~~the~~ method ~~according to claim 1.~~ **for prevention of priority inversion without priority promotion**

mechanism when ~~of~~ executing processes with different priorities in a multiprocessing environment comprising execution of a low priority process and a high priority process and optionally at least one intermediate priority process having a priority between that of the low priority process and the high priority process, where the high priority process and the low priority process share a given resource, the medium comprising:

code for temporarily raising an effective priority of the low priority process when the low priority process is going to use the shared resource, wherein the raising of the effective priority comprises executing/assigning an additional process accessing the shared resource on behalf of the low priority process where the additional process has a priority higher than the at least one intermediate priority process in the multiprocessing environment.

14. Cancelled.

15. Cancelled.